

DOCUMENT CONSTRAINT DESCRIPTORS OBTAINED FROM USER**SIGNALS INDICATING ATTRIBUTE-VALUE RELATIONS**

This application claims priority under 35 U.S.C. §120 from copending International Applications PCT/IB98/00757 and PCT/IB98/00758, both filed 23 April 1998, with respect to all shared subject matter. International Application PCT/IB98/00757 in turn claimed priority from Great Britain Application No. 9708175.6, filed 23 April 1997, and International Application PCT/IB98/00758 in turn claimed priority from Great Britain Application No. 9708172.3, filed 23 April 1997. WO-98/48359, the published version of International Application PCT/IB98/00757, and WO-98/48361, the published version of International Application PCT/IB98/00758, are both incorporated herein by reference in entirety.

Field of the Invention

The invention relates to techniques that obtain constraints for documents.

Background

Andreoli, J.-M., Borghoff, U.M., Pareschi, R., and Schlichter, J.H., "Constraint Agents for the Information Age", *Journal of Universal Computer Science*, Vol. 1, No. 12, December 1995, pp. 762-789, describe constraint-based knowledge brokers which are concurrent agents that use signed feature constraints to represent partially specified information and can flexibly cooperate in the management of distributed knowledge.

1 Andreoli et al. disclose an operation named "scope-splitting", which relies
 2 on the use of negation. Under scope-splitting, a broker can split its scope,
 3 creating two brokers. In contrast with a basic feature constraint (BFC), which
 4 cannot include negation or disjunction, a signed feature constraint (SFC) is
 5 composed of a positive part and a list of negative parts, both of which are basic
 6 feature constraints. If the scope of a broker is represented by an SFC and the
 7 scope is split by a BFC, the two resulting split scopes can both be represented
 8 by SFCs. In an example, a database of documents by non-American authors
 9 about art can be split by a constraint "books written after 1950" into art books
 10 written after 1950 but not by an American author and art documents not
 11 authored by an American but not books subsequent to 1950.

12 Andreoli et al. also disclose techniques for solving SFCs. Constraint
 13 satisfaction over BFCs is defined by conditional rewrite rules, as is conventional.
 14 Given an SFC, its positive component is first normalized by the algorithm for
 15 BFCs. If the result is a contradiction, the SFC is unsatisfiable. But otherwise,
 16 the normalized positive component is inserted into each of the negative
 17 components, which are then normalized by the algorithm for BFCs. If a resulting
 18 negative component has a contradictory normal form, it is eliminated, but if it has
 19 a tautological normal form, the SFC is unsatisfiable. The SFC is thus satisfiable
 20 if and only if its normal form is not reduced to a contradiction. Andreoli et al.
 21 disclose an implementation in which the SFC solver is realized as a list-
 22 transforming algorithm with additional checks for constraint satisfaction.

1 Andreoli et al. also disclose that a set of initial brokers can be provided,
2 each with predefined scope. In processing requests, new brokers and agent
3 specialists are cloned to handle a subset of their parent scope. In responding to
4 follow-on requests, answers from existing specialists can be used, and the scope
5 splitting mechanism avoids redundant work. Complex requests require
6 interactions with many other agents and information stored in the network. In
7 large information networks, such as the World-Wide Web, the reuse of
8 generated and already collected information is especially important.

9 **Summary of the Invention**

10 The invention addresses problems that arise in obtaining constraints for
11 documents.

12 With widespread availability of new electronic sources of information, such
13 as e-mail, Internet access, and on-line information repositories, the number of
14 electronic documents available to a computer user is multiplying. Documents
15 can also be built dynamically by accessing and combining information existing
16 over distributed sources. Hierarchical mark-up languages such as SGML can be
17 used to define document templates that can be dynamically filled in with
18 heterogeneous components. All of the various types of documents can in turn
19 be given a degree of permanence by storing them in document management
20 systems, thus entering them into a normal document lifecycle despite their
21 differences.

1 The Document Management Alliance (DMA) has attempted to provide an
2 industry standard for search, retrieval, storage, and conversion of electronic
3 documents on heterogeneous document management systems.

4 Knowledge broker techniques as described by Andreoli et al., above, can
5 be used to perform search and retrieval of electronic documents in accordance
6 with the DMA standard and other such standards. The Andreoli et al. techniques
7 employ feature constraints that can be built from atomic constraints, either sorts
8 or features. A sort is a unary relation, expressing a property of a single entity,
9 while a feature is a binary relation expressing a property linking two entities. The
10 Andreoli et al. technique uses signed feature constraints, composed of both a
11 positive part and a list of negative parts, where the positive part and each
12 negative part, before negation, is a basic feature constraint that includes neither
13 negation nor disjunction.

14 A typical user, however, has difficulty formulating sorts and features that
15 will produce a desired constraint. As a result, techniques like that of Andreoli et
16 al. are only useful for expert users who understand the logic of sorts and
17 features and can formulate a set of sorts and features for a desired constraint.

18 This and related problems are referred to herein as the "constraint
19 production problems".

20 The invention alleviates constraint production problems by providing
21 techniques that obtain document constraint descriptors for documents from user
22 signals. A document constraint descriptor includes information about a set of

1 one or more constraints that documents could satisfy. Instead of requiring the
 2 user to provide a set of sorts and features, the techniques allow the user to
 3 provide attribute-value relations, which are relatively easy for typical users to
 4 provide. The techniques then convert the attribute-value relations to logical
 5 relations such as sorts and features from which a constraint descriptor can be
 6 obtained.

7 The new techniques can be implemented in a method for obtaining
 8 document constraint descriptors from user signals. The method can receive user
 9 signals indicating a set of attribute-value relations that can apply to documents.
 10 The method can use the user signals to obtain logical relations equivalent to the
 11 attribute-value relations. The method can then use the logical relations to obtain
 12 a document constraint descriptor defining a set of one or more constraints
 13 equivalent to the logical relations.

14 The method can be performed with a machine that includes user interface
 15 circuitry, through which the machine can receive a series of user signals. The
 16 machine can be a portable computing device with a touchscreen or a keyboard.
 17 Or the machine can be a fixed computing device with one or more of a
 18 touchscreen, keyboard, and mouse. Or the machine can be a multifunction
 19 device with a scanner, which can scan an image-bearing portable medium such
 20 as a form to produce electronic signals, in turn used to obtain the user signals;
 21 the form can include a field with a human readable indication of an attribute and
 22 an area a user can mark to indicate a set of values of the attribute.

1 The user interface circuitry can include display circuitry for presenting
2 images and selection circuitry the user can operate to indicate items in the
3 images. The document constraint descriptor can be stored in memory and the
4 method can present an image that includes an item representing the descriptor,
5 receive a user signal indicating the item, and, in response, obtain the stored
6 descriptor.

7 The method can solve the set of constraints to obtain a solution and use
8 the solution to obtain document references indicating electronic documents in a
9 repository accessible through a network. The method can present an image that
10 includes items representing the document references and, in response to a user
11 signal indicating one of the items, can retrieve the item's electronic document. A
12 portion of the electronic document can be displayed or the electronic document
13 can be printed.

The new techniques can also be implemented in a machine. In general, the machine can include a processor and user interface circuitry for providing user signals to the processor. The processor can operate as described above, receiving user signals indicating a set of attribute-value relations applicable to documents, use the user signals to obtain logical relations, and use the logical relations to obtain a document constraint descriptor.

20 The new techniques are advantageous because they provide convenient
21 ways for ordinary users to produce document constraint descriptors. The

1 descriptors can be used to specify search requests, answers to requests, and
2 the state of retrieval agents.

3 The following description, the drawings, and the claims further set forth
4 these and other aspects, objects, features, and advantages of the invention.

5 **Brief Description of the Drawings**

6 Fig. 1 is a schematic circuit diagram showing a network through which
7 constraint descriptors could be transferred.

8 Fig. 2 is a schematic diagram showing the scope defined by a constraint.

9 Fig. 3 is a schematic version of an image presented by a fixed computing
10 device in response to user signals indicating attribute-value relations.

11 Fig. 4 is a schematic flow chart of operations in obtaining a constraint
12 descriptor from user signals provided through user interface circuitry, such as in
13 response to images as in Fig. 3.

14 Fig. 5 is a schematic diagram showing features of a form that can be
15 marked to provide user signals indicating attribute-value relations.

16 Fig. 6 is a schematic flow chart of operations performed in obtaining a
17 constraint descriptor from user signals provided through a scanner, such as with
18 a form as in Fig. 5.

19 Fig. 7 is a schematic flow chart of operations performed in using a
20 constraint to retrieve document references and the documents they indicate.

1 Fig. 8 is a schematic version of an image presented by a fixed computing
2 device presenting a list of items representing document references, as in box
3 s94 in Fig. 7.

4 Fig. 9 is a schematic version of an image presented by a fixed computing
5 device showing selected items from the list in Fig. 8 after transformation into
6 HTML format, as in box s95 in Fig. 7.

7 Fig. 10 is a schematic version of an image presented by a fixed computing
8 device presenting in more detail a single item from the list in Fig. 8.

9 **Detailed Description**

10 A. Conceptual Background

11 The following definitions are helpful in understanding the broad scope of
12 the invention, and the terms defined below have the indicated meanings
13 throughout this application, including the claims.

14 A “processor” or “processing circuitry” is a component of circuitry that
15 responds to input signals by performing processing operations on data and by
16 providing output signals. A processor may include one or more central
17 processing units or other processing components. A processor can be a general
18 purpose processor or a special purpose processor.

1 A "portable computing device" is a device that includes at least a
2 processor and input/output circuitry and can be moved from place to place
3 without difficulty.

4 A "fixed computing device" is a device that includes at least a processor
5 and input/output circuitry and is not a portable computing device.

6 A processor or processing circuitry performs an operation or a function
7 "automatically" when it performs the operation or function independent of
8 concurrent human intervention or control.

9 A "user interface" or "user interface circuitry" is circuitry that can provide
10 signals from a user. A user interface can, for example, include display circuitry
11 for presenting images to a user and selection circuitry for providing user signals
12 indicating items in the images. A user interface could include a scanner that
13 produces electronic signals that include user signals, such as user markings in a
14 field of a form.

15 Any two components are "connected" when there is a combination of
16 circuitry that can transfer signals from one of the components to the other. For
17 example, two components are "connected" by any combination of connections
18 between them that permits transfer of signals from one of the components to the
19 other.

20 A "network" is a combination of circuitry through which a connection for
21 transfer of data can be established between machines. An operation

1 "establishes a connection over" a network if the connection does not exist before
2 the operation begins and the operation causes the connection to exist.

3 Any two components "communicate" when signals are transferred from
4 one of the components to the other. Therefore, "communicating circuitry" is
5 circuitry in a component that provides communication between the component
6 and one or more other components. In addition to circuitry that provides direct
7 connection or connection through a network, communicating circuitry can include
8 transmitters and receivers for electromagnetic waves or other signals that do not
9 require connections.

10 A "data packet" is an item of data that communicating circuitry can use to
11 communicate, by converting a data packet into signals at a sending component
12 and by extracting a data packet from signals at a receiving component.

13 In a very broad sense, a "document" is an object from which information
14 can be extracted that can be understood by a human, possibly after decoding or
15 other processing of the object. An "electronic document" is a document in an
16 electronic form, such as when being stored in memory circuitry or when being
17 transmitted between machines by communicating circuitry, even though the
18 medium of communication may not itself be electronic.

19 A "document repository" is a component within which electronic
20 documents may be stored for subsequent access and retrieval.

1 A "document reference" is an item of data that can be used to access a
2 specific document stored by a document repository, and may be said to
3 "indicate" or "identify" the document. Web URLs and other unique identifiers of
4 documents are examples of document references.

5 To "obtain" or "produce" an item of data is to perform any combination of
6 operations that begins without the item of data and that results in the item of
7 data. To obtain a first item of data "based on" a second item of data is to use the
8 second item to obtain the first item.

9 The notions of "constraint" and "satisfy" are related: A constraint is a
10 condition that, when met, is satisfied. A "constraint that documents can satisfy"
11 is therefore a condition that could be met by a document. A constraint can be a
12 logical combination of constraints, such as a conjunction of a set of
13 subconstraints, in which case the constraint "includes" the subconstraints. For
14 example, constraints that documents can satisfy may be expressed as logical
15 combinations of simpler constraints such as attribute-value relations, where each
16 attribute-value relation is between an attribute that a document could have and a
17 set of at least one value of the attribute. A constraint is "inconsistent" if it cannot
18 be met because of its logical structure; if inconsistency of a constraint can be
19 determined from logical structure, it is unnecessary to search or check whether a
20 document can be found that meets the constraint--no document could possibly
21 meet it. A constraint that is not inconsistent is "satisfiable" even though it may
22 not in fact be satisfied by any stored document.

1 Similarly, a set of constraints is equivalent to a set of logical relations only
 2 if the constraints are only satisfied when the logical relations are evaluated as
 3 true and the constraints are only not satisfied when the logical relations are
 4 evaluated as false.

5 A "solution" of a constraint or a set of constraints is an item of data that
 6 indicates whether the constraint or set of constraints is inconsistent or satisfiable
 7 and, if satisfiable, indicates a less redundant version that is equivalent to the
 8 constraint or set of constraints. In this context, the solution is "equivalent" to the
 9 constraint or set of constraints if the solution can only be satisfied if the
 10 constraint or set of constraints is satisfied and vice versa.

11 An operation "solves" a constraint or a set of constraints if it obtains a
 12 solution of the constraint or set of constraints.

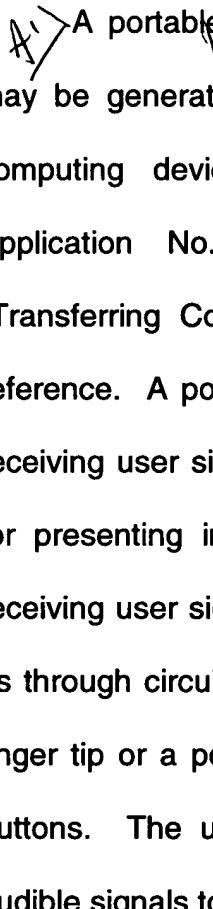
13 B. System

14 The invention can be implemented using conventional computing devices
 15 with communication provided by conventional computer network technology,
 16 such as a local area network (LAN), a wide area network (WAN), or other
 17 appropriate technology. The invention has been successfully implemented using
 18 conventional Web browser software, such as Netscape Navigator, to provide
 19 cross-platform communication and document transfer over the Internet. The
 20 implementation employs a type of constraint descriptors referred to herein as
 21 "feature constraints", described in greater detail below.

1 Fig. 1 illustrates schematically network 21, in which the Internet transfers
 2 feature constraints between machines 22, 24, and 26. Each machine could be
 3 any conventional computing device connected to the Internet, such as a PC
 4 running Windows, a Mac running MacOS, or a minicomputer or other machine
 5 running Unix. Other system configurations could be employed, such as those
 6 described by Flynn et al., above, and other network configurations could be
 7 employed, including those described in EP-A-772,857 and US-A-5,692,073. In
 8 general, each of the computing devices connected to network 21 can include
 9 user interface circuitry for receiving user signals, a processor whose operations
 10 are responsive to the user signals, and memory for storing data. The user
 11 interface circuitry can, for example, include display circuitry such as circuitry to
 12 present images on a CRT, LCD, or other display device. The user interface
 13 circuitry can also include selection circuitry for receiving signals indicating items
 14 in images presented by the display circuitry, such as circuitry to receive signals
 15 from a keyboard, mouse, touchscreen sensor, joystick, or other such device.

16 In response to a request from a user at receiving machine 22, a document
 17 stored on sending machine 26 can be retrieved and sent over the Internet to
 18 receiving machine 22, via one or more intermediate machines 24. As is well
 19 known, a document accessible through the Web can be retrieved using as a
 20 unique identifier its Web URL, as described by Flynn et al., above. As further
 21 described by Flynn et al., additional devices of various types can be connected
 22 to network 21, including scanners, printers, copiers, and multifunction devices
 23 capable of scanning, printing, faxing, etc., described, for example, in

1 EP-A-741,487. Each machine connected to network 21 can also be equipped
 2 with appropriate hardware and software for communication with portable
 3 computing devices, such as conventional hardware and software for
 4 communication with personal digital assistants (PDAs), handheld PCs, pocket or
 5 wristwatch computers, or other portable computers.

6  A portable computing device and techniques by which search requests
 7 may be generated in response to a data packet from a user of a portable
 8 computing device are disclosed in copending, coassigned U.S. Patent
 9 Application No. 09/XXX,XXX (Attorney Docket No. R/97005), entitled
 10 "Transferring Constraint Descriptors for Documents", incorporated herein by
 11 reference. A portable computing device can include user interface circuitry for
 12 receiving user signals. The user interface circuitry can include display circuitry
 13 for presenting images on a small bitmap screen and selection circuitry for
 14 receiving user signals indicating items in images presented on the screen, such
 15 as through circuitry that senses a position at which the screen is touched by a
 16 finger tip or a pointer and circuitry for receiving signals provided through push
 17 buttons. The user interface circuitry can also include circuitry for providing
 18 audible signals to the user through a tone generator.

19 Portable computing devices, and some or all fixed computing devices
 20 connected to network 21 can be equipped for infrared communication or for
 21 wireless communication at other wavelengths, such as by well known radio
 22 technology. For example, data packets transmitted between a portable
 23 computing device and other devices, such as data packets encoding information

1 enabling document retrieval, can conform to the physical and link layer formats
 2 (IrLAP) described in the industry standard Infrared Data Association (IrDA)
 3 specification, version 1.0, or subsequent versions, as is well known in the art.
 4 For this purpose, a portable computing device can have 19.2 Kb/s bi-directional
 5 IR communication circuitry for transmitting and receiving through a diode
 6 transmitter/receiver.

7 A portable computing device could also include communication circuitry
 8 for providing a wired or docking link to other portable computing devices or to
 9 fixed computing devices, using conventional techniques.

10 A portable computing device can include a conventional microprocessor
 11 that presents images on its screen, that receives user signals through the user
 12 interface circuitry and through its transmitter/receiver, and that provides signals
 13 to other computing devices through its transmitter/receiver. The microprocessor
 14 can be connected to conventional memory circuitry for storage of data.

15 As will be understood from the description below, the microprocessor
 16 could receive user signals indicating attribute-value relations such as sets of
 17 values for one or more attributes of a document, could obtain equivalent logical
 18 relations, could then obtain an equivalent document constraint descriptor for a
 19 set of constraints, and could store the constraint descriptor in memory. Then, in
 20 response to further user signals, the microprocessor could encode the document
 21 constraint descriptor in a data packet and provide the data packet for
 22 transmission to another device. For example, the microprocessor could present

1 an icon with a description of the constraint descriptor and, in response to a user
2 signal indicating the icon, could transmit a data packet encoding the constraint
3 descriptor to a fixed computing device to initiate a search for documents
4 satisfying the set of constraints.

5 The microprocessor could operate in various other ways, some of which
6 are mentioned below.

7 C. Knowledge Brokers and Feature Constraints

8 ~~Although the invention could be implemented in various ways, the~~
9 ~~invention has been successfully implemented by programming computing~~
10 ~~devices to employ knowledge brokers and feature constraints as described by~~
11 ~~Andreoli et al., above. A demonstration of a prototype can be viewed at~~
12 ~~<http://www.xrce.xerox.com/research/ct/projects/cbkb/home.html>. This section~~
13 ~~reviews relevant aspects of knowledge brokers and feature constraints.~~

14 Brokers are software agents that can process knowledge search requests.
15 Knowledge is taken here to be any piece of electronic information intended to be
16 publicly accessible. Different, possibly distributed, information sources are
17 assumed to be available, from a simple file in a user's directory to a database
18 local to a site, up to a wide area information service (WAIS) on the Internet, for
19 example.

20 When receiving a request, a broker may have sufficient knowledge to
21 process it, or may need to retrieve more knowledge. For that purpose, it

1 releases sub-requests, aimed at other brokers. Thus, knowledge retrieval is
2 achieved by the collaboration of all the brokers, which are alternatively service
3 providers processing requests and clients of these services generating
4 sub-requests. The infrastructure required to support such collaboration, and the
5 way knowledge is stored locally within each broker can be understood from
6 Andreoli, J.-M., Borghoff, U., and Pareschi, R., "The Constraint-Based
7 Knowledge Broker Model: Semantics, Implementation and Analysis", *Journal of*
8 *Symbolic Computation*, Vol. 21, No. 4, 1996, pp. 635-676, incorporated herein by
9 reference. The following discussion addresses rather the knowledge
10 manipulations occurring within each broker.

11 In order to collaborate, the brokers must at least understand each other.
12 This can be achieved by formulating all requests and all answers to requests in a
13 common language, even if the brokers may perform local translations. Logic
14 provides an adequate language for such a purpose. A request can be expressed
15 by a pair $\langle x, P \rangle$ where x is a logical variable and P a logical formula involving x ,
16 meaning "Retrieve knowledge objects x such that the property expressed by
17 formula P holds". Interestingly, an answer to such a request can be expressed in
18 the same formalism, i.e. a pair $\langle x, Q \rangle$ meaning "There exists a knowledge object
19 x satisfying the property expressed by formula Q ". The requirement here is that
20 P must be a logical consequence of Q , so that the answer contains at least as
21 much knowledge as the request. Moreover, the same logical formalism can be
22 used to capture the scope of a broker, i.e. the area of knowledge it is concerned
23 with: A broker with scope $\langle x, R \rangle$ means "I am not capable of retrieving

1 knowledge objects x which do not satisfy the property expressed by formula R' .
2 In many situations, the scope of a broker may vary, because it is specialized or,
3 on the contrary, expands its capacities, either externally or due to the knowledge
4 retrieval process itself.

5 In other words, logic provides a common language in which requests,
6 answers, and scopes can all be expressed. Brokers then perform logical
7 operations on these three components. The most important logical operation,
8 from which all the others can be reconstructed, is satisfiability checking, i.e.
9 deciding whether some object could satisfy the property expressed by a formula,
10 or, on the contrary, whether it is intrinsically contradictory. Unfortunately, it is
11 well known that this operation, for full classical logic, is not algorithmic, i.e. it is
12 provably impossible to write a program which implements it and always
13 terminates. Given this limitation, a great deal of research in knowledge
14 representation has been focused on identifying fragments of classical logic in
15 which satisfiability is algorithmically decidable. The trade-off here is between
16 expressive power and tractability: The empty fragment, for example, is obviously
17 tractable, but it is not very expressive.

18 The most popular fragment which emerged is known as "feature
19 constraints". The satisfiability problem in this case is also known as "feature
20 constraint solving".

21 As is known, feature constraints can be built from atomic constraints that
22 are either sorts or features. A sort is a unary relation, expressing a property of a

1 single entity. For example, $P:\text{person}$ expresses that an entity P is of sort
 2 person. A feature is a binary relation expressing a property linking two entities.
 3 For example, $P:\text{employer} \rightarrow E$ expresses that entity P has an employer, which
 4 is an entity E . Apart from sorts and features, most feature constraint systems
 5 also allow built-in relations such as equality and inequality, and such relations
 6 are also referred to herein as "built-in predicates" or "built-in constraints".

7 The full fragment of feature constraints, where the atomic components
 8 mentioned above are allowed to be combined by all the logical connectives
 9 (conjunction, disjunction, negation and quantifiers), although very expressive, is
 10 hardly tractable. A subfragment called "basic feature constraints" (BFC) has
 11 been considered, where negation and disjunction are simply forbidden. Efficient
 12 constraint solving algorithms have been proposed for this sub-fragment.
 13 However, a drawback is that the complete absence of negation puts strong
 14 limitations on the kind of operations a knowledge broker may wish to perform.

15 Brokers can use a powerful operation, referred to as "scope-splitting",
 16 which relies on the use of negation. Indeed, a broker may wish to split its scope,
 17 specified by a pair $\langle x, P \rangle$ according to a criterion expressed by a formula F , thus
 18 creating two brokers with scope $P \wedge F$ and $P \wedge \neg F$. Thus, a broker in charge of
 19 bibliographic information may wish to split its scope into two new scopes: "books
 20 written after 1950", which can be represented by a BFC that includes two feature
 21 constraints and a built-in constraint

1 X
2 X : book
3 X : year -> Y
4 Y > 1950,

5 and its complement, i.e. "books written before 1950 or documents which are not
6 books"; this latter scope cannot be expressed using BFC, because negation and
7 disjunction cannot be dispensed with. It has been discovered that the scope
8 splitting operation is useful in many situations, for example to implement brokers
9 capable of memorizing and reusing information gathered during their lifetime. A
10 broker can, for example, use, on the one hand, a fragment of feature constraints,
11 called "signed feature constraints" (SFC), which allows limited use of negation,
12 precisely capable of expressing the kind of split scope mentioned above, and, on
13 the other hand, an efficient constraint solving method for SFC.

14 A signed feature constraint is composed of a positive part and a list of
15 negative parts, both of them being basic feature constraints. For example, the
16 following signed feature constraint

17 P
18 + P : person,
19 P : employer-> E,
20 E : "Xerox"
21 - P : nationality-> N,
22 N : "American"
23 - P : spouse-> P'
24 P' : person
25 P' : employer-> E'
26 E' : "Xerox"

27 specifies a Xerox employee who is not American and is not married to another
28 Xerox employee.

1 This SFC can be represented graphically as in Fig. 2. The round boxes
2 denote the entities (logical variables), the sort relations (unary) are represented
3 by dashed arrows labeled by the name of the sort in a square box, the feature
4 relations (binary) are represented by plain arrows labeled by the name of the
5 feature in a square box. Built-in predicates (not present in the example) could be
6 represented by rhombuses. The positive part of the SFC is contained in the top
7 box and marks the distinguished entity of the scope (P in the example) by a
8 double round box. The negative parts of the SFC are contained in the lower
9 boxes in gray.

10 The main interest of SFCs comes from the following property: If the
11 scope of a broker is represented by an SFC e_0 , and this scope is split by a BFC
12 e , then the two resulting split scopes e^+ , e^- are both SFCs.

13 Indeed, e^+ can be obtained by merging the positive part of e_0 with the
14 BFC e , and e^- can be obtained by extending e_0 with a new negative part
15 containing e alone. For example, assume a broker in charge of a bibliographical
16 database containing various documents (books, videos etc.) about art, but not
17 authored by an American. The database can be represented by the SFC

18 X
19 +X : topic-> T
20 T : "Art"
21 -X : author-> A
22 A : nationality-> N
23 N : "American"
24

25 The SFC may be split by the constraint "books written after 1950", expressed by
26 the BFC

```

1 X
2 X : book
3 X : year-> Y
4 Y > 1950
5

```

6 The resulting scopes are simply

```

7 X
8 +X : book
9 X : topic-> T
10 X : year-> Y
11 T : "Art"
12 Y > 1950
13 -X : author-> A
14 A : nationality-> N
15 N : "American"
16

```

17 i.e. "Art books written after 1950 but not by an American author" and

```

18 X
19 +X : topic-> T
20 T : "Art"
21 -X : author-> A
22 A : nationality-> N
23 N : "American"
24 -X : book
25 X : year-> Y
26 Y > 1950

```

27 i.e. "Art documents not authored by an American but not books subsequent to
28 1950".

29 Most constraint systems make a number of assumptions on the nature of
30 sorts and features, called the axioms of the systems. These axioms are crucial
31 to the satisfiability algorithm, since they determine when a feature constraint is
32 contradictory and when it is satisfiable.

33 For the purpose of simplicity, the implementation disclosed here makes
34 use of a slight variant of the basic axiom system used in Aït-Kaci, H. et al., "A
35 Feature-Based Constraint-System for Logic Programming with Entailment",

1 *Theoretical Computer Science*, Vol. 122, 1994, pp. 263-283, although it will be
2 appreciated by persons skilled in the art that the principles of the method apply
3 to other sets of axioms as well.

4 1. Features are functional: This means that if two pairs of entities
5 which are constrained by the same feature have the same first term, they also
6 have the same second term. For example, it can be considered that the feature
7 spouse is functional (within a specific cultural setting), meaning that a person
8 cannot have two spouses: If, for a person x , we have $x:\text{spouse} \rightarrow Y$ and
9 $x:\text{spouse} \rightarrow Z$, then the entities Y and Z coincide (i.e. denote the same
10 person). Other systems allow multi-valued features.

11 2. Sorts are disjoint: this means that no entity can be of two distinct
12 sorts. For example, a book is not a person: We cannot have an entity x with
13 $x:\text{book}$ and $x:\text{person}$. Other systems consider hierarchies of sorts where
14 some entities can have multiple sorts as long as they have a common
15 denominator in the hierarchy.

16 3. There is a distinguished subset of sorts, called "value" sorts, so that
17 no two distinct entities can be of the same value sort. Traditional basic elements
18 (strings, numbers, etc.) are typical value sorts: For example, the string "Xerox"
19 or the number 1950 are value sorts. Value sorts are not allowed to have
20 features: This is the only axiom connecting sorts and features. Other systems
21 consider more refined connections between sorts and features.

1 4. There is a distinguished built-in binary predicate, equality, with the
2 traditional congruence axioms (which involve sorts and features). The axioms
3 describing all the other built-in predicates are assumed to contain no mention of
4 sorts and features.

5 ^{A3} These axioms are formally written in section A: Axioms in the Appendix at
6 the end of this specification. They form a theory T .

7 ^{A4} Based on this axiom system, a set of SFCs can be solved by a constraint
8 satisfaction process as follows:

9 First, it is assumed that satisfiability over built-in predicates is decidable.
10 This means that there is an algorithm which, given a formula F using only built-in
11 predicates (F is also called a built-in constraint), can decide whether F is a logical
12 consequence of the theory T (written $\vdash_T F$).

13 ^{A5} Constraint satisfaction over BFCs is defined by a set of conditional rewrite
14 rules over BFCs (section B.1 of the Appendix) which have the following
15 properties

16 ^{A6} (a) The system of rules is convergent and hence defines a "normal
17 form" for BFCs. This can be shown in a classical way by proving that the system
18 is "Church-Rosser" (critical pairs converge) and "Noetherian" (the size of the
19 terms strictly decrease by rewriting).

20 (b) A BFC is satisfiable if and only if its normal form is not reduced to
21 the contradiction. One implication can be proved by showing that rewrite steps

1 preserve satisfiability. The reverse implication can be proved by displaying a
2 model that satisfies BFCs whose normal form is not reduced to the contradiction.

3 Thus the rewrite rules describe the steps of a constraint satisfaction
4 algorithm. This algorithm always terminates because the system of rewrite rules
5 is convergent. It is to be noted that the definition of the rules relies on
6 satisfiability tests of built-in constraints, which have been assumed decidable.
7 This means that the algorithm is modular and can accommodate any kind of
8 built-in constraints as long as a proper built-in constraint satisfaction algorithm is
9 provided.

10 Rewrite rules for a constraint satisfaction algorithm can be implemented in
11 a naive way in some symbolic language like Lisp or Prolog, or can be optimized,
12 taking into account the properties of the specific built-in constraints which are
13 used.

14 ¹⁷ The algorithm for constraint satisfaction over SFCs (section B.2 of the
15 Appendix) can informally be described as follows. Given an SFC, its positive
16 component is first normalized by the algorithm for BFCs. If the result is a
17 contradiction, the whole SFC is unsatisfiable. Otherwise, the normalized positive
18 component is inserted in each of the negative components, which are then
19 normalized by the algorithm for BFCs. If a resulting negative component has a
20 contradictory normal form, it is eliminated, and if it has a tautological normal form
21 the whole SFC is unsatisfiable. The normal form for SFCs thus obtained has the
22 following property:

1 An SFC is satisfiable if and only if its normal form is not reduced to the
2 contradiction. A non-contradictory normal form is thus a solution of the SFC.

3 D. Transactions

4 Figs. 3-10 illustrate several transactions that can be performed in the
5 current implementation.

6 The processor of a device can obtain a document constraint descriptor
7 from user input signals in a number of ways.

8 Fig. 3 shows an image presented by display circuitry of a fixed computing
9 device while a user is entering a query, e.g. "books or articles after 1990 in which
10 the title contains 'constraints' but does not contain 'internet'". The image
11 includes boxes any of which the user can select by mouse inputs, after which the
12 user can type or complete an element of the query in the selected box. For
13 example, within a dedicated window, the user can enter a query into main query
14 entry box 30, which can be implemented using conventional techniques. Box 30
15 includes boxes 31 and 32 either of which the user can select by mouse signals,
16 after which the user can type into or complete an element of a query in the
17 selected box. As shown, box 31 includes the element "books/articles" of a
18 query, while box 32 includes the element "internet".

19 The image in Fig. 3 also includes buttons 33 that the user can select by
20 mouse inputs to select an attribute of a document, such as "title", or a constraint
21 operator applicable to an attribute, such as "contains not". Additional buttons 34

1 and 36 allow the user to restart, add to, edit, build up, or otherwise modify a
2 query.

3 Each element of the query is added to the current specification of the
4 query, and the image in Fig. 3 also includes box 37 that contains the current
5 specification. The image also includes button 38, which the user can select to
6 launch a search based on the current specification of the query.

7 Fig. 4 shows operations that can be performed by the processor of a
8 device that presents images as in Fig. 3 in obtaining a document constraint
9 descriptor. In box s41, the processor displays the knowledge broker (KB) main
10 query window as shown in Fig. 3 and prompts the user to indicate that a query
11 can be entered. In box s42, the processor receives a series of user signals
12 through user interface circuitry that indicate keyed-in query elements. As shown
13 in Fig. 3, the query elements can be attribute-value relations such as "title
14 contains 'constraints'", indicating that the document attribute "title" has a value
15 that includes the word "constraints", or "date after 90", indicating that the
16 document attribute date has a year value greater than 1990. Several other types
17 of attribute-value relations are described in relation to Fig. 5 below.

18 In box s43, the processor adds each query element from box s42 to an
19 existing list of query elements in the current specification. The processor also
20 updates the display by presenting an image in which the "Current Specification"
21 in box 37 includes the added query element.

1 to the processor of a portable computing device, using a keyboard or a
2 touchscreen user interface on which lists of items are displayed and can be
3 navigated or selected using scrolling and control buttons. Where the screen of a
4 device is too small for such techniques, members of a stored set of items of data
5 could be accessed in the manner described in EP-A-733,964.

6 Fig. 5 shows query sheet 50 that a user can mark to indicate a query.
7 Query sheet 50 could be printed on a sheet of paper or on another suitable
8 image-bearing portable medium that can be scanned by the scanner of a
9 multifunction device or a scanner connected to a fixed computing device. Query
10 sheet 50 might alternatively be presented on a display with a touchscreen or
11 other circuitry capable of sensing marking movements.

12 Query sheet 50 is a form that includes fields 51, 52, 53, and 54, each for
13 indicating a set of values for a document attribute. As indicated by a human-
14 readable title for each field, the user can indicate the type of a document in field
15 51, the author in field 52, the date in field 53, and the topic in field 54. Each field
16 also includes human-readable cues on how its blank areas should be completed
17 to indicate sets of values for attributes. Fields could be added, on the same
18 sheet or other sheets, for various additional document attributes or for other
19 information that could be used in requesting a search. A description of query
20 sheet 50 can be stored, enabling a machine to apply appropriate recognition to
21 checks, characters, or other marks handwritten, typed, or otherwise made in
22 each field by a user, according to conventional techniques such as mark
23 sensing, optical character recognition, and handwriting recognition.

1 In field 52, the user can write characters in boxes 55 to indicate the family
 2 name and given name of the author. The user can use a wildcard character "*" to
 3 indicate that a name may be completed by any combination of characters.
 4 The user can mark the "Not" box next to either name to indicate that the
 5 handwritten value should be excluded from the search results. In the illustrated
 6 example, the user has indicated that the value for the author's family name
 7 should be "JOBS" and the value for the author's given name should begin with
 8 the three letters "STE".

9 In field 53, the user can write characters in boxes 56 to indicate year,
 10 month, and day of the document. The user can also mark one of the boxes in
 11 the upper portion of field 53 to indicate whether the date should be equal to,
 12 after, or before the indicated date. In the illustrated example, the user has
 13 indicated that the value for the document's date should be after June 1993.

14 In field 54, the user can write characters in boxes 57 to indicate two
 15 topics, again using the wildcard character if appropriate. Field 54 also includes
 16 boxes that can be marked to indicate whether either topic should be excluded
 17 from the search and to indicate whether the topics should be combined by "And"
 18 or "Or". In the illustrated example, the user has indicated that the value for a
 19 topic of the document should begin with "client . . ." and that the values for topics
 20 of the document should not include "mobile".

1 In field 51, the user can mark boxes, such as boxes 58, to indicate the
2 type of a document. In the illustrated example, the user has indicated that the
3 value for the document's type should be any type other than "journal".

4 Fig. 6 shows operations that can be performed by the processor of a
5 device that receives user signals based on a scanned image of a query sheet as
6 in Fig. 5 in obtaining a document constraint descriptor. In box s60, the query
7 sheet is scanned and the processor stores the image data file in memory for
8 processing. The image data file can, for example, be a bitmap of the image.

9 In box s61, the processor analyzes the image data file, applying
10 appropriate criteria to determine, for each location of a check box, such as boxes
11 58, whether the data indicate that a check has been made in the box. The
12 processor can store data indicating which check boxes have been checked.

13 In box s62, the processor can use the information from the check boxes
14 and also images extracted from other boxes, such as boxes 55, 56, and 57, to
15 determine query elements associated with each of fields 51, 52, 53, and 54. If a
16 query element includes an image of a box that has been marked with a
17 character, handwriting recognition or OCR can be performed as necessary to
18 obtain values for the query element, in box s63. The processor can add each
19 query element obtained in boxes s62 and s63 to an existing list of query
20 elements in the current specification.

1 Then the processor can convert each query element in the existing list to
2 a logical relation, in box s64. This can be accomplished in the same manner
3 described above for box s44 in Fig. 4

4 Once the logical relations are obtained, the processor can automatically
5 compile a signed feature constraint from the logical relations, in box s65, as in
6 box s45 in Fig. 4.

7 The processor could also present an image prompting the user to enter a
8 query identifier, such as a short query name, through the user interface circuitry.
9 The processor can receive the name from the user, or can automatically
10 generate a default name if no name is received. The processor can then store
11 data associating the stored feature constraint with its name and with data
12 defining an icon for the feature constraint. The processor can also present an
13 image that includes the icon and name of the feature constraint.

14 The user interface circuitry of a portable computing device could present
15 an image that includes icons with document names to represent document
16 references such as stored Web URLs. In addition, the image could include icons
17 and short query names, representing stored document constraint descriptors.

18 AS } A document constraint descriptor could be transferred between two
19 computing devices in the manner described in copending, coassigned U.S.
20 Patent Application No. 09/XXX,XXX (Attorney Docket No. R/97005), entitled
21 "Transferring Constraint Descriptors for Documents", incorporated herein by
22 reference.

1 Fig. 7 illustrates operations performed by the processor of a computing
2 device in using a document constraint descriptor, such as a feature constraint, to
3 retrieve document references and in displaying or printing documents. The
4 operations could be performed, for example, by the processor of a fixed
5 computing device such as a conventional PC, Mac, or workstation or by a
6 multifunction device or by a printer with an appropriate user interface.

7 In box s91, the processor receives a feature constraint from a device,
8 such as from a device that obtained the feature constraint in accordance with
9 Fig. 4 or Fig. 6, above. The processor can receive the feature constraint in a
10 data packet from a portable computing device or in any other appropriate way.

11 In box s92, the processor receives further user signals requesting a
12 search for documents satisfying the feature constraint. The user signals can
13 again be received in any appropriate way, such as by presenting an image that
14 includes an item representing the feature constraint and receiving a user signal
15 selecting the item.

16 In response, the processor can solve the feature constraint using the
17 techniques described above for solving basic feature constraints and signed
18 feature constraints. If compilation in box s45 or box s65 completely solves the
19 equivalent constraint, no further solution is necessary in box s92, but if
20 compilation in box s45 or box s65 merely changes format or the like, it is
21 necessary to perform all the remaining computation necessary to obtain a
22 solution. Therefore, solving the constraint in box s92 can be thought of as

1 completing the solution process that was begun by compiling in box s45 or box
2 s65. The solution process could be divided between compilation and solving in
3 many different ways, and the two operations could be at least partially
4 redundant.

5 If the processor obtains a solution, the solution can be used to formulate a
6 search request, which the processor can then provide in a call to search engine
7 routines it also executes. In general, the search engine routines can in turn call
8 remote search engines, such as through the Internet, and any appropriate
9 combination of local and remote search operations can be employed.

10 In box s93, the processor executes the search engine routines and uses
11 the search request formulated in box s92 to perform a search of all appropriate
12 repositories on a network to which the computing device is connected.
13 Alternatively, the search could be performed on any appropriate subset of the
14 repositories. The search could include providing versions of the search request
15 to other search engines on the network. Where necessary, the search engine
16 can perform a brokering process that breaks down the search request from box
17 s92 into subrequests as described in Andreoli, J.-M., Borghoff, U., and Pareschi,
18 R., "The Constraint-Based Knowledge Broker Model: Semantics, Implementation
19 and Analysis", *Journal of Symbolic Computation*, Vol. 21, No. 4, 1996, pp. 636-
20 676, incorporated herein by reference. As will be understood, the brokering
21 process may include scope splitting, specialization of brokers, and solution of
22 constraints equivalent to subrequests.

1 The search engine routines return a list of "hits", i.e. document references
 2 such as Web URLs identifying documents satisfying the feature constraint. In
 3 box s94, the processor retrieves the list of hits and presents an image that
 4 includes information about the hits. Fig. 8 illustrates an example of an image
 5 that could be presented, with a window in which each hit is represented by an
 6 item in the form of a line of text. Each hit's line of text includes the hit's number
 7 and a brief description of the document indicated by the hit, such as the
 8 document's title.

9 The user can provide input signals requesting information about one or
 10 more identified individual hits. In response, in box s95, the processor can
 11 present one or more further images with information about the identified
 12 individual hits, such as by presenting a hit with expanded details about the
 13 document, by presenting document information converted into HTML format, or
 14 by presenting a version of the document itself downloaded from the repository
 15 that contains it.

16 Fig. 9 illustrates an example of an image that could be presented in box
 17 s95, in which the information about each hit has been converted into HTML
 18 format. For each hit, the display information can include author name, http URL,
 19 information source, reference, and title.

20 Fig. 10 illustrates another example of an image that could be presented in
 21 box s95, in which a more complete set of attributes of one hit's document is
 22 included. In Fig. 10, the displayed values for some of the attributes are not

1 explicitly shown, but are shown as URLs that provide links to pages that contain
2 information related to those attributes.

3 The user can also provide input signals requesting that a hit's document
4 be printed or be sent to a user specified printer for printing. In response, in box
5 s96, the processor can download the document and print it on the user's default
6 printer or on a user specified printer.

7 D. Variations

8 The implementations described above could be varied in numerous ways
9 within the scope of the invention.

10 The implementation described above has been successfully executed
11 using machines specified above, but implementations could be executed on
12 other machines.

13 The implementation described above has been successfully executed
14 using software described above, but various other software could be used,
15 developed for a wide variety of programming environments and platforms. For
16 example, techniques other than knowledge brokers and feature constraints could
17 be used.

18 The implementation described above obtains document constraint
19 descriptors that are signed feature constraints obtained in specified ways using
20 logical relations equivalent to attribute-value relations, but the invention could be
21 implemented to obtain other types of document constraint descriptors, including

1 basic feature constraints and built-in constraints, and with logical relations and
2 constraints obtained in various other ways from attribute-value relations. For
3 example, the mapping from attribute-value relations to logical relations could be
4 performed algorithmically or using any of a wide variety of data structures, and
5 constraints could be obtained from logical relations using any of a wide variety of
6 compilation and solution techniques.

7 The implementation described above can transfer constraints between
8 specified types of computing devices using specified communication techniques
9 such as IrDA standard data transfer and the Internet, but the invention could be
10 implemented to transfer constraints between a wide variety of different
11 computing devices and using any of a wide variety of communication techniques.
12 For example, the invention could be implemented using devices that are all
13 connected to a network, or it could be implemented using devices that cannot
14 communicate through a network, but can only communicate through
15 electromagnetic waves such as IR or radio waves, or it could be implemented
16 using any combination of such devices.

17 In the implementation described above, the computing devices have user
18 interface circuitry that includes specified types of devices, such as displays,
19 keyboards, touchscreens, buttons, mice, but the invention could be implemented
20 with any suitable kind of user interface circuitry.

21 The implementation described above presents specific types of images in
22 which items include icons and names or titles, but the invention could be

1 implemented with or without presentation of images, and the images presented
 2 could take any appropriate form, with or without icons and with or without names
 3 or titles. The images could, in addition, be presented through a paper user
 4 interface using printed check boxes on paper or the like.

5 The implementation described above employs URLs as document
 6 references, but document references could take any appropriate form. For
 7 example, the World Wide Web Consortium (W3C) defines uniform resource
 8 names (URNs) that could be used.

9 The implementation described above uses search engine routines to find
 10 documents satisfying a constraint, and a wide variety of search engines using
 11 various search techniques could be used to find such documents.

12 In the implementation described above, documents are retrieved for
 13 display or printing, but documents or knowledge from documents could instead
 14 be retrieved for other purposes, such as to generate a new document.

15 The implementation described above mentions several specific attributes
 16 of documents with specific types of values, but a wide variety of other document
 17 attributes could be used, with various types of values. Furthermore, the
 18 implementation described above treats attributes or features of documents as
 19 independent, and could be applied even to attributes or features with trivial
 20 dependencies that can be ignored, but a different approach might be required to
 21 obtain optimal results with attributes or features that have complex
 22 dependencies.

1 In the implementation described above, specific acts are performed that
 2 could be omitted or performed differently. For example, in Fig. 4, a logical
 3 relation could be obtained and feature constraint compilation could be performed
 4 after each query element is added to the list, or it could only be performed when
 5 requested by a user.

6 In the implementation described above, acts or operations are performed
 7 in an order that could be modified in many cases. For example, in Fig. 7,
 8 individual hits could be displayed immediately when obtained from the search
 9 engine rather than first displaying a list of hits.

10 The implementation described above uses currently available computing
 11 techniques, but could readily be modified to use newly discovered computing
 12 techniques as they become available.

13 E. Applications

14 The invention can be applied to document information retrieval and
 15 distribution, such as in a system that employs the Internet. The system can
 16 include a combination of portable and fixed computing devices.

17 F. Miscellaneous

18 The invention has been described in relation to software implementations,
 19 but the invention might be implemented with specialized hardware.

1 Although the invention has been described in relation to various
2 implementations, together with modifications, variations, and extensions thereof,
3 other implementations, modifications, variations, and extensions are within the
4 scope of the invention. The invention is therefore not limited by the description
5 contained herein or by the drawings, but only by the claims.

660207-948246